

# **Применение псевдоразметки для распознавания речи**

**+ дополнения про авторегрессионные модели**

**Павел Богомолов, 24.10.2023**

# План

## Авторегрессионные модели

- Проблемы расхождения обучения и инференса
- Teacher forcing
- Minimum WER training
- Shallow fusion alternatives

# План

## Псевдоразметка

- Стандартный пайплайн обучения
- Простейшая псевдоразметка (pseudo-labeling)
- Resnet: аугментации
- Итеративная псевдоразметка
- slimIPL, экспоненциальное сглаживание учителя
- Объединение wav2vec 2.0 и псевдолейблов
- Софт-лейблы и связь с knowledge distillation

# Авторегрессионные модели

## Расхождение обучения и инференса

- Обучение:
  - ???
- Инференс:
  - ???

# Авторегрессионные модели

## Расхождение обучения и инференса

- Обучение:
  - Подаем все токены референса до  $i$ -го, предсказываем  $i$ -й
- Инференс:
  - Модель сама генерирует все, начиная с первого токена
  - Делаем beam-search, поддерживаем  $N$  гипотез
  - Подмешиваем скор языковой модели

# Авторегрессионные модели

## Teacher forcing

- При обучении мы знаем итоговую транскрипцию, поэтому можем ее подавать на вход модели
- В реальности модель сама выдает токены и их же получает на вход
- Можно генерировать при обучении, но это долго и может сбивать модель при плохом исходном качестве
- Компромисс 1: использовать teacher forcing с некоторой вероятностью, которую мы понижаем со временем (**scheduled sampling**)

# Авторегрессионные модели

## Teacher forcing

- При обучении мы знаем итоговую транскрипцию, поэтому можем ее подавать на вход модели
- В реальности модель сама выдает токены и их же получает на вход
- Можно генерировать при обучении, но это долго и может сбивать модель при плохом исходном качестве
- Компромисс 1: использовать teacher forcing с некоторой вероятностью, которую мы понижаем со временем (**scheduled sampling**)
- Компромисс 2: заменять часть токенов во входе декодера на случайные

# Авторегрессионные модели

## mWER

- При обучении модель старается сгенерировать лучший ответ
- В реальности мы делаем beam search и стараемся ранжировать гипотезы, при этом наша метрика — WER
- Можно ли оптимизировать WER?

Word Error Rate (WER)

- **S** — substitutions
- **I** — insertions
- **D** — deletions
- **N** — reference words

$$\text{WER} = \frac{S + I + D}{N}$$



# Авторегрессионные модели

## mWER

In order to minimize word error rates on test data, we consider as our loss function, the *expected number of word errors over the training set*:

$$\mathcal{L}_{\text{werr}}(\mathbf{x}, \mathbf{y}^*) = \mathbb{E}[\mathcal{W}(\mathbf{y}, \mathbf{y}^*)] = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \mathcal{W}(\mathbf{y}, \mathbf{y}^*) \quad (4)$$

# Авторегрессионные модели

## mWER

In order to minimize word error rates on test data, we consider as our loss function, the *expected number of word errors over the training set*:

$$\mathcal{L}_{\text{werr}}(\mathbf{x}, \mathbf{y}^*) = \mathbb{E}[\mathcal{W}(\mathbf{y}, \mathbf{y}^*)] = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \mathcal{W}(\mathbf{y}, \mathbf{y}^*) \quad (4)$$

$$\mathcal{L}_{\text{werr}}^{\text{N-best}}(\mathbf{x}, \mathbf{y}^*) = \sum_{\mathbf{y}_i \in \text{Beam}(\mathbf{x}, N)} \hat{P}(\mathbf{y}_i|\mathbf{x}) \left[ \mathcal{W}(\mathbf{y}_i, \mathbf{y}^*) - \widehat{W} \right]$$

Where,  $\hat{P}(\mathbf{y}_i|\mathbf{x}) = \frac{P(\mathbf{y}_i|\mathbf{x})}{\sum_{\mathbf{y}_i \in \text{Beam}(\mathbf{x}, N)} P(\mathbf{y}_i|\mathbf{x})}$ , represents the distribution re-normalized over just the N-best hypotheses, and  $\widehat{W}$  is the average number of word errors over the N-best hypotheses, which is applied as a form of variance reduction, since it does not affect the gradient.

# Авторегрессионные модели

## mWER

In order to minimize word error rates on test data, we consider as our loss function, the *expected number of word errors over the training set*:

$$\mathcal{L}_{\text{werr}}(\mathbf{x}, \mathbf{y}^*) = \mathbb{E}[\mathcal{W}(\mathbf{y}, \mathbf{y}^*)] = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) \mathcal{W}(\mathbf{y}, \mathbf{y}^*) \quad (4)$$

$$\mathcal{L}_{\text{werr}}^{\text{N-best}}(\mathbf{x}, \mathbf{y}^*) = \sum_{\mathbf{y}_i \in \text{Beam}(\mathbf{x}, N)} \hat{P}(\mathbf{y}_i|\mathbf{x}) \left[ \mathcal{W}(\mathbf{y}_i, \mathbf{y}^*) - \widehat{W} \right]$$

Where,  $\hat{P}(\mathbf{y}_i|\mathbf{x}) = \frac{P(\mathbf{y}_i|\mathbf{x})}{\sum_{\mathbf{y}_i \in \text{Beam}(\mathbf{x}, N)} P(\mathbf{y}_i|\mathbf{x})}$ , represents the distribution re-normalized over just the N-best hypotheses, and  $\widehat{W}$  is the average number of word errors over the N-best hypotheses, which is applied as a form of variance reduction, since it does not affect the gradient.

$$\mathcal{L}^{\text{N-best}} = \sum_{(\mathbf{x}, \mathbf{y}^*)} \mathcal{L}_{\text{werr}}^{\text{N-best}}(\mathbf{x}, \mathbf{y}^*) + \lambda \mathcal{L}_{\text{CE}}$$

# Авторегрессионные модели

## Shallow fusion

- Авторегрессионные модели выучивают языковое распределение внутри себя
- ^ почему это проблема при shallow fusion?

# Авторегрессионные модели

## Shallow fusion

Shallow Fusion:

$$\hat{y} = \arg \max_y (\log P_{\text{E2E}}(y|x) + \lambda \log P_{\text{ELM}}(y) + \beta |y|)$$

# Авторегрессионные модели

## Shallow fusion

### Density Ratio

- hybrid

$$\hat{y} = \arg \max_y P_{\text{AM}}(x|y) P_{\text{ELM}}(y)$$

$$P_{\text{RNNT}}(x|y) = \frac{P_{\text{RNNT}}(y|x) P_{\text{RNNT}}(x)}{P_{\text{RNNT}}(y)}$$

$$\hat{y} = \arg \max_y \frac{P_{\text{RNNT}}(y|x) P_{\text{RNNT}}(x)}{P_{\text{RNNT}}(y)} P_{\text{ELM}}(y) = \arg \max_y \frac{P_{\text{RNNT}}(y|x)}{P_{\text{RNNT}}(y)} P_{\text{ELM}}(y)$$

$$P_{\text{RNNT}}(y) \approx P_{\text{ILM}}(y)$$

$$\hat{y} = \arg \max_y (\log P_{\text{RNNT}}(y|x) - \log P_{\text{ILM}}(y) + \log P_{\text{ELM}})$$

$$\hat{y} = \arg \max_y (\log P_{\text{RNNT}}(y|x) + \lambda_0 \log P_{\text{ILM}}(y) + \lambda_1 \log P_{\text{ELM}} + \beta|y|)$$

**ILM** — Internal Language Model: отдельно тренируем neural-LM на train-транскрипциях

# Аutoreгрессионные модели

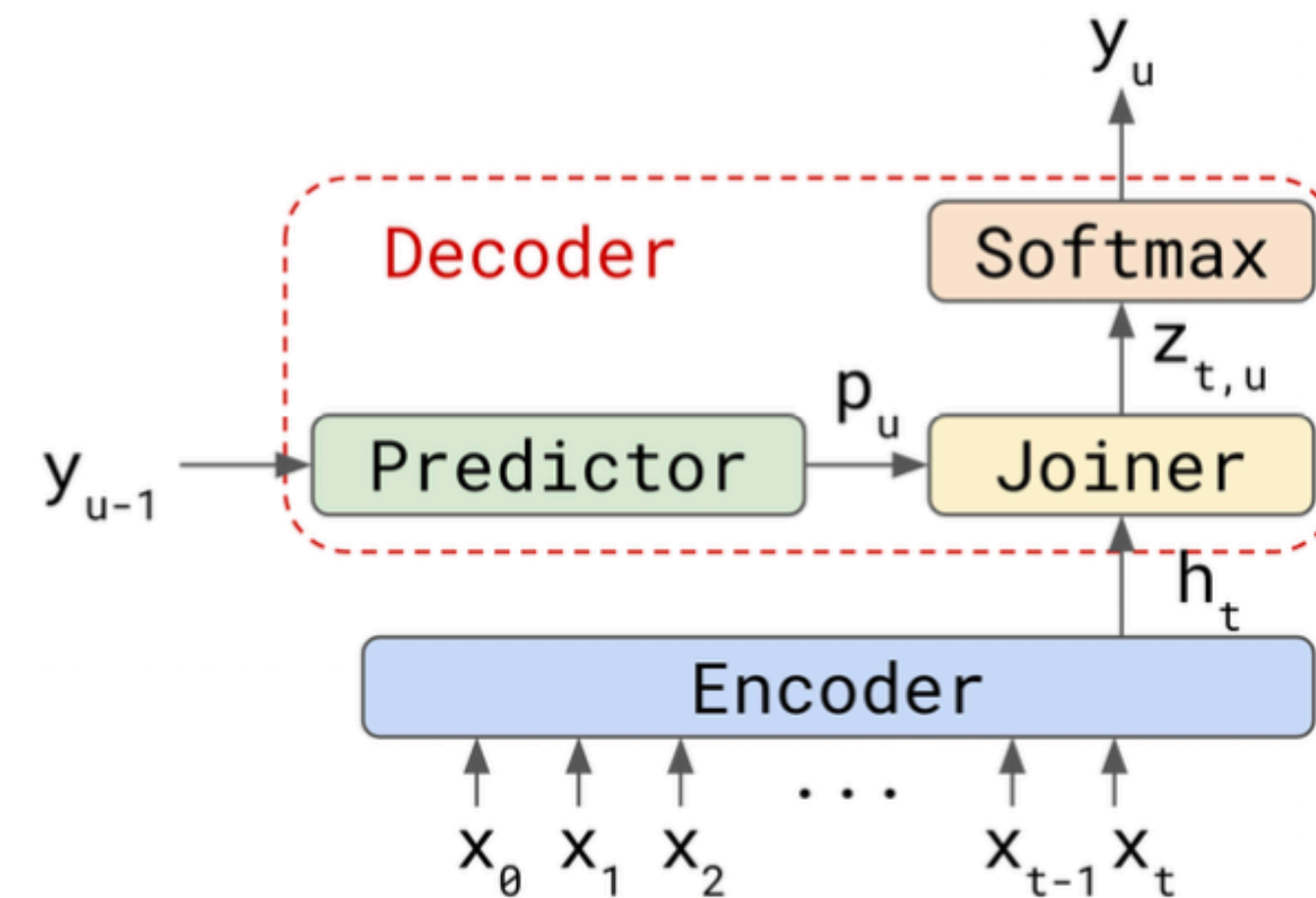
## Shallow fusion

ILME

$$h_{t,u}^{\text{joint}} = \tanh(W_1^{\text{joint}} h_t^{\text{enc}} + W_2^{\text{joint}} h_u^{\text{pred}})$$

$W_2^{\text{joint}} \rightarrow 0 \Rightarrow \text{CTC-model}$

$h_t^{\text{enc}} \rightarrow 0 \Rightarrow \text{Internal Language Model Estimation}$



$$\log P_{\text{ILM}} = \sum_{u=0}^U \log \text{softmax } J(h_u^{\text{pred}}, \mathbf{0})$$

# Авторегрессионные модели

## Shallow fusion

LODR: low-order density ratio method

были статьи, в которых показывалось, что **ILM** в RNNT использует языковую информацию с малым контекстом в прошлое:

- Prediction Network в RNN-T не совсем корректно считать языковой моделью
- Ограничение контекста Prediction Network на основе LSTM не ухудшает качество

при этом в Density Ratio используется полноценная языковая модель с большим контекстом. Авторы предлагают low-order density ratio: использовать слабую bi-gram LM



# Псевдоразметка

# Стандартный пайплайн обучения

- У нас есть размеченные данные  $L$ : пары  $(x, y)$ , где  $x$  - звук,  $y$  - текст
- Мы обучаем акустическую модель, используя CTC или другой лосс
- Применяем модель: beam-search decoding с LM

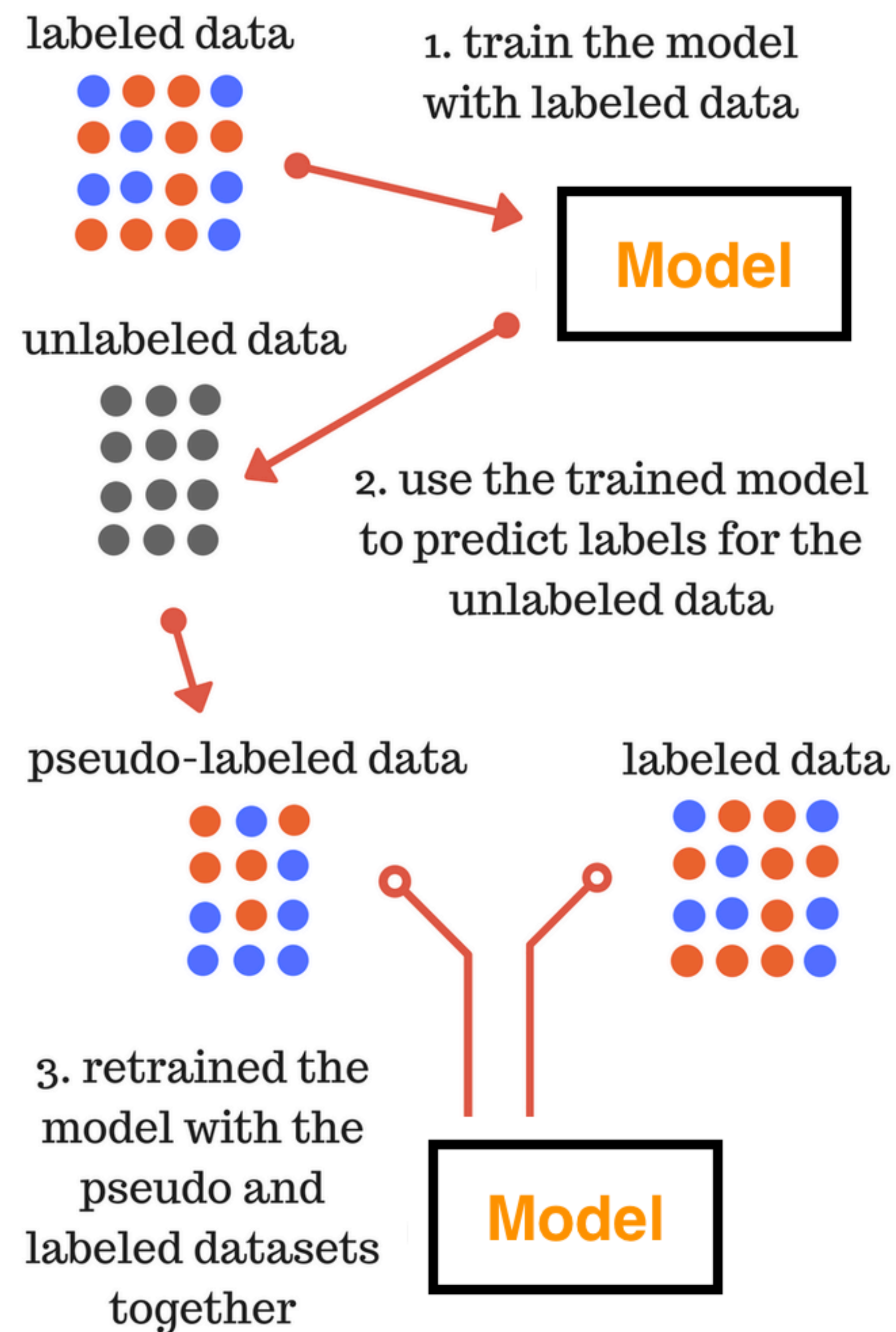
# Стандартный пайплайн обучения

- Проблема: разметка зачастую дорогая, либо мы вообще не можем ее делать (например, это приватные данные)
- В итоге зачастую количество размеченных данных сильно меньше, чем неразмеченных, но последние мы никак не используем
- Что делать?

# Псевдо-разметка

## Простейший вариант

- Обучим модель  $M_0$  на размеченных данных  $L$
- Возьмем наш неразмеченный датасет  $U$ , содержащий звук  $x_U$
- Применим к нему модель:  $y_U = M_0(x_U)$ , используя beam-search и LM
- Обучим на  $L$  и  $(x_U, y_U)$  новую модель  $M_1$
- Применяем модель: beam-search decoding с LM



# Псевдо-разметка

Почему это работает?

# Псевдо-разметка

## Почему это работает?

- Интуитивно кажется, что модель не может выучить ничего нового из своих же предсказаний
- На деле:
  - Языковая модель сдвигает предсказание в сторону себя
  - Beam search позволяет получить лучшее предсказание
  - За счет того, что мы берем hard лейблы (не вероятности), это уже получается не совсем то же самое, что простой выход модели
  - Если добавить аугментации при обучении, усложняем M1 задачу еще сильнее

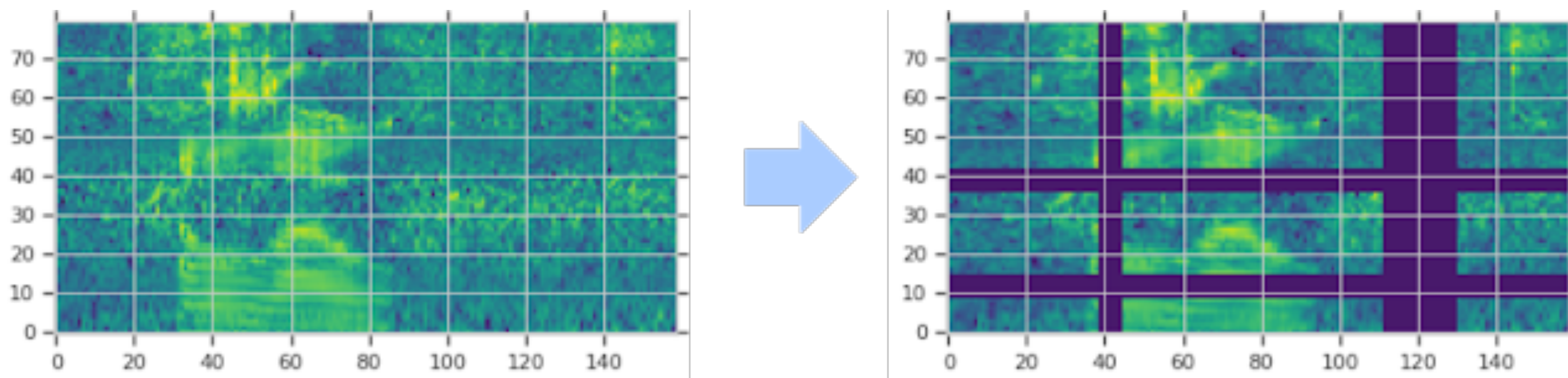
# Самые популярные аугментации

## (Бонус)



# Самые популярные аугментации (Бонус)

- Ускорение/замедление записи
- Добавление шумов
- СресAugment: зануляем некоторые фреймы и частоты – самая популярная аугментация в статьях





# Итеративная псевдоразметка

## "Iterative Pseudo-Labeling for Speech Recognition" (Facebook)

- Повторить процесс, обучив модель M2, M3 и т.д.
- Бонус: стартовать с обученных весов, чтобы ускорить процесс

---

### Algorithm 1: Iterative pseudo-labeling

---

**Data:** Labeled data  $L = \{x_i, y_i\}_{i=1}^l$ , Unlabeled data  $U = \{x'_j\}_{j=1}^u$

**Result:** Acoustic model  $p_\theta$

Initialize  $p_\theta$  by training on only labeled data  $L$ ;

**repeat**

1. Draw a subset of unpaired data  $\tilde{U} \in U$ ;

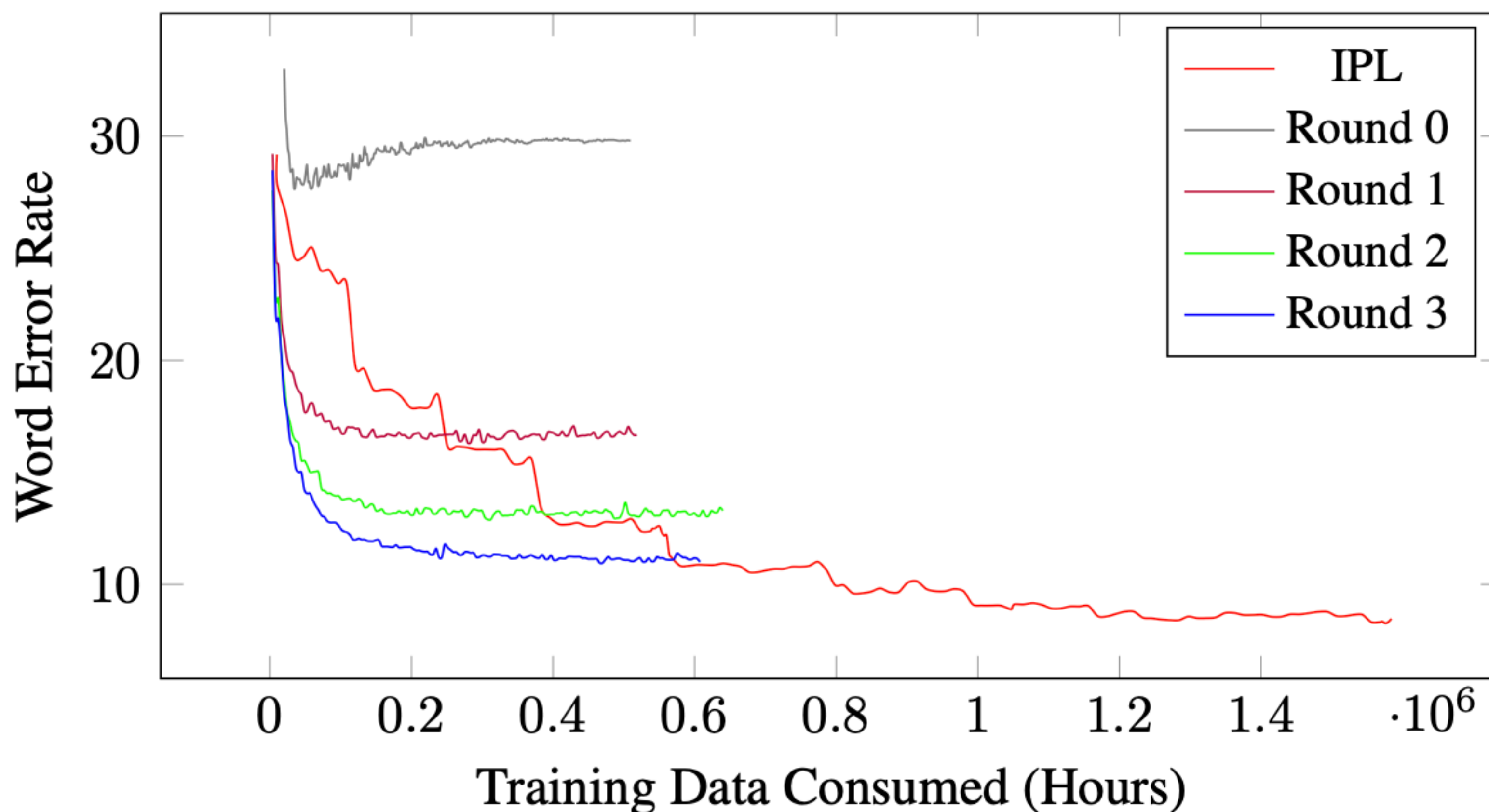
2. Apply  $p_\theta$  and decoding with LM to the subset  $\tilde{U}$  to generate  $\hat{U} = \{(x, \hat{y}) | x \in \tilde{U}\}$ ;

3. Fine tune  $p_\theta$  on  $L \cup \hat{U}$  with data augmentation;

**until** *convergence or maximum iterations are reached*;

# Итеративная псевдо-разметка

"Iterative Pseudo-Labeling for Speech Recognition" (Facebook)



# Итеративная псевдо-разметка

"Iterative Pseudo-Labeling for Speech Recognition" (Facebook)

Table 3: WER of greedy path on dev-other for IPL and training from scratch for multiple rounds. 4-gram  $LS \setminus LV$  LM is used for pseudo-labels generation.

Data		# Rounds of PL				IPL
Labeled	Unlabeled	0	1	2	3	
LS-100	LS-860	27.76	17.1	15.8	15.09	10.69
LS-100	LS + LV	27.76	16.3	12.9	10.95	7.90
LS-960	LV-54K	7.31	5.00	4.69	4.57	4.12

# Итеративная псевдоразметка

## Проблемы

- Начинаем расходиться со временем
- Псевдолейблы от LAS-моделей плохо работают:
  - модель может выдавать слишком короткие транскрипции (*почему?*)
  - модель может заикливаться
- Каждый раз регенерировать предсказания может быть дорого

# Итеративная псевдоразметка

## Решения

- Начинаем расходиться со временем  
=> Храним псевдолейблы от старых версий модели (по сути получаем псевдолейблы от ансамбля всех версий)
- Псевдолейблы от LAS-моделей плохо работают  
=> Фильтруем предсказания
- Каждый раз регенерировать предсказания может быть дорого  
=> Убираем LM и beam-search decoding  
Почему оно до сих пор работает? Аугментации!

# Про расходимость

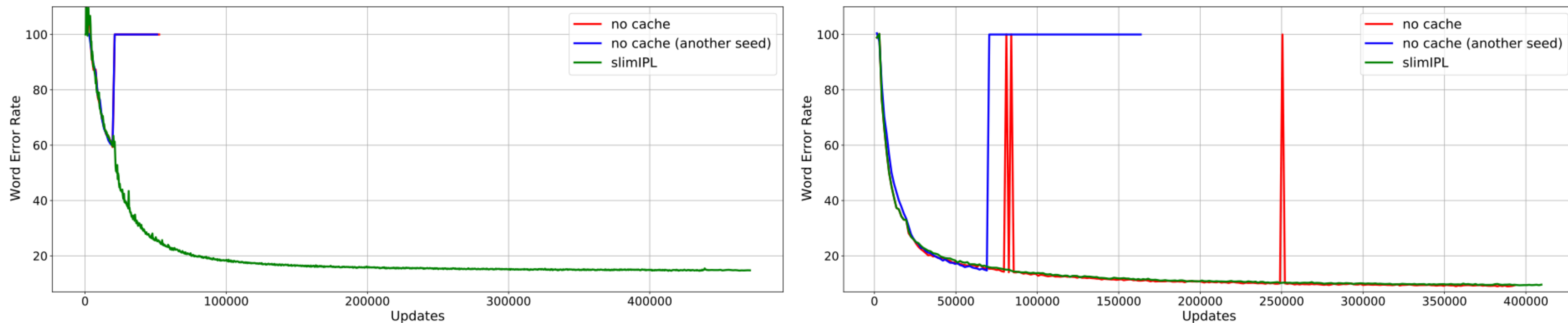
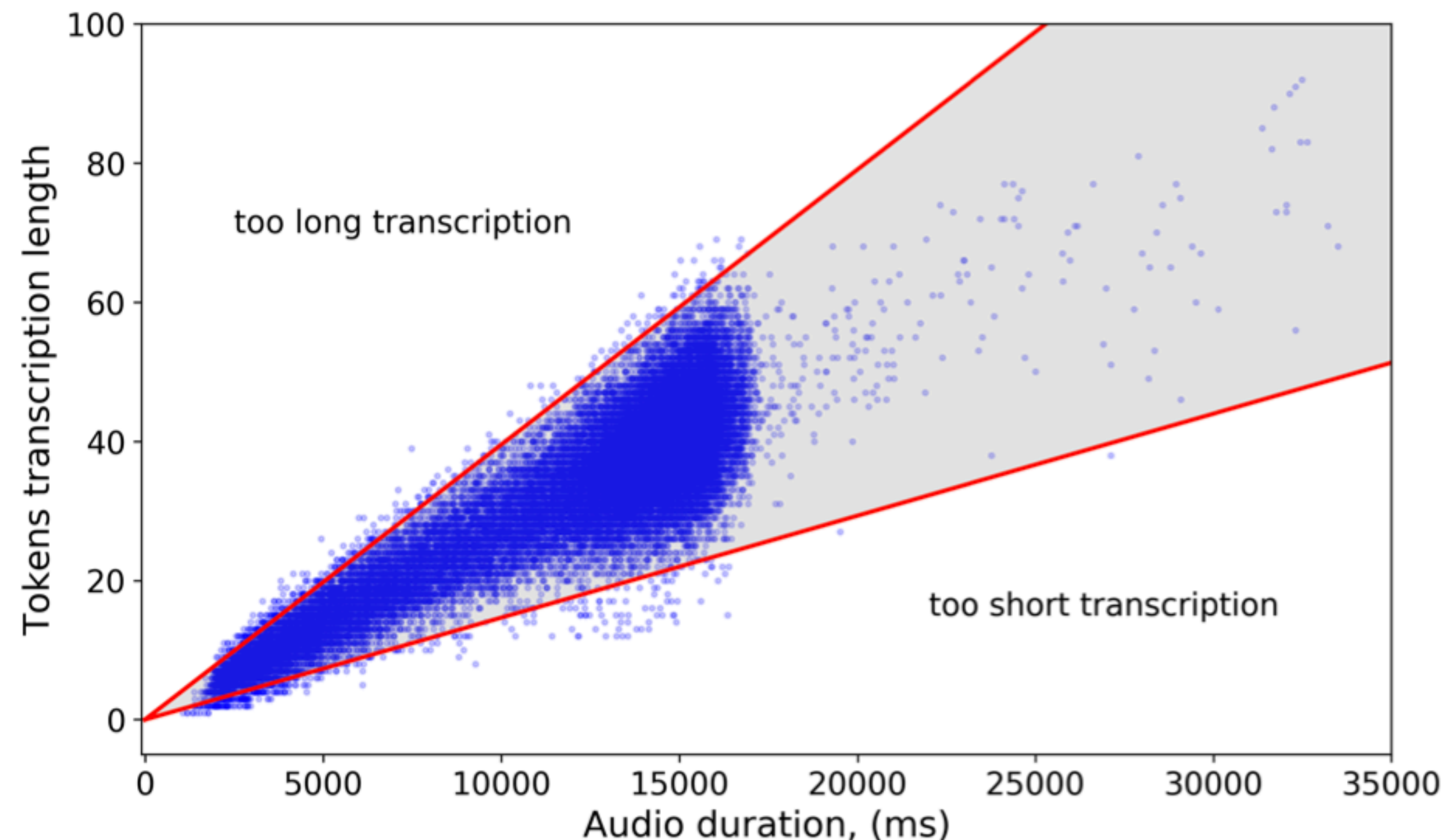


Figure 1: Learning curves on *dev-other* for models trained on LL-10/LS-960 (left) and LS-100/LS-860 (right). slimIPL models refer to baseline models (grey) from Table 3.



# Фильтрация

- Можем брать скор языковой модели (Google)
- Можем построить эвристику: смотреть на соотношение длин звука и таргета, отсекаать выбросы (Facebook, актуально для LAS-моделей)
- Выкидывать предсказания, где случилось заикливание (Facebook, LAS)
- Фильтрацию со временем можно ослаблять, это увеличит датасет => можно ослаблять аугментации/регуляризацию
- (А можно наоборот усиливать аугментацию, усложняя задачу для модели)



# Итеративная псевдоразметка

## slimIPL

---

**Algorithm 1:** slimIPL

---

**Data:** labeled  $L = \{\mathbf{x}_i, \mathbf{y}_i\}$  and unlabeled  $U = \{\mathbf{x}_j\}$

**Result:** Acoustic model  $\mathcal{M}_\theta$

1. Train  $\mathcal{M}_\theta$  on  $L$  with augmentation for  $M$  updates;

2. **while** *cache is not full at size  $C$*  **do**

- Draw a random batch from  $\mathbf{x} \in U$ ;
- Generate its PL  $\hat{\mathbf{y}}$  by  $\mathcal{M}_\theta$  following Eq.(1);
- Store  $\{\mathbf{x}, \hat{\mathbf{y}}\}$  into the cache;
- Train  $\mathcal{M}_\theta$  on  $L$  with augmentation for 1 update;

**end**

3. Decrease model's  $\mathcal{M}_\theta$  dropout;

**repeat**

4. Train  $\mathcal{M}_\theta$  on  $L$  with augmentation for  $N_L$  updates;

5. **for**  $N_U$  *updates* **do**

- Draw a random batch  $B = \{\mathbf{x}, \hat{\mathbf{y}}\}$  from the cache;
- **With probability**  $p$ ,  $B$  is removed from cache and a new pair of random batch  $\mathbf{x}' \in U$  and its PL  $\hat{\mathbf{y}}'$  generated by  $\mathcal{M}_\theta$  is added in;
- Apply augmentation to batch  $B$  and make an optimization step to update  $\mathcal{M}_\theta$ .

**end**

**until** *convergence or maximum iterations are reached*;

---



# Итеративная псевдоразметка

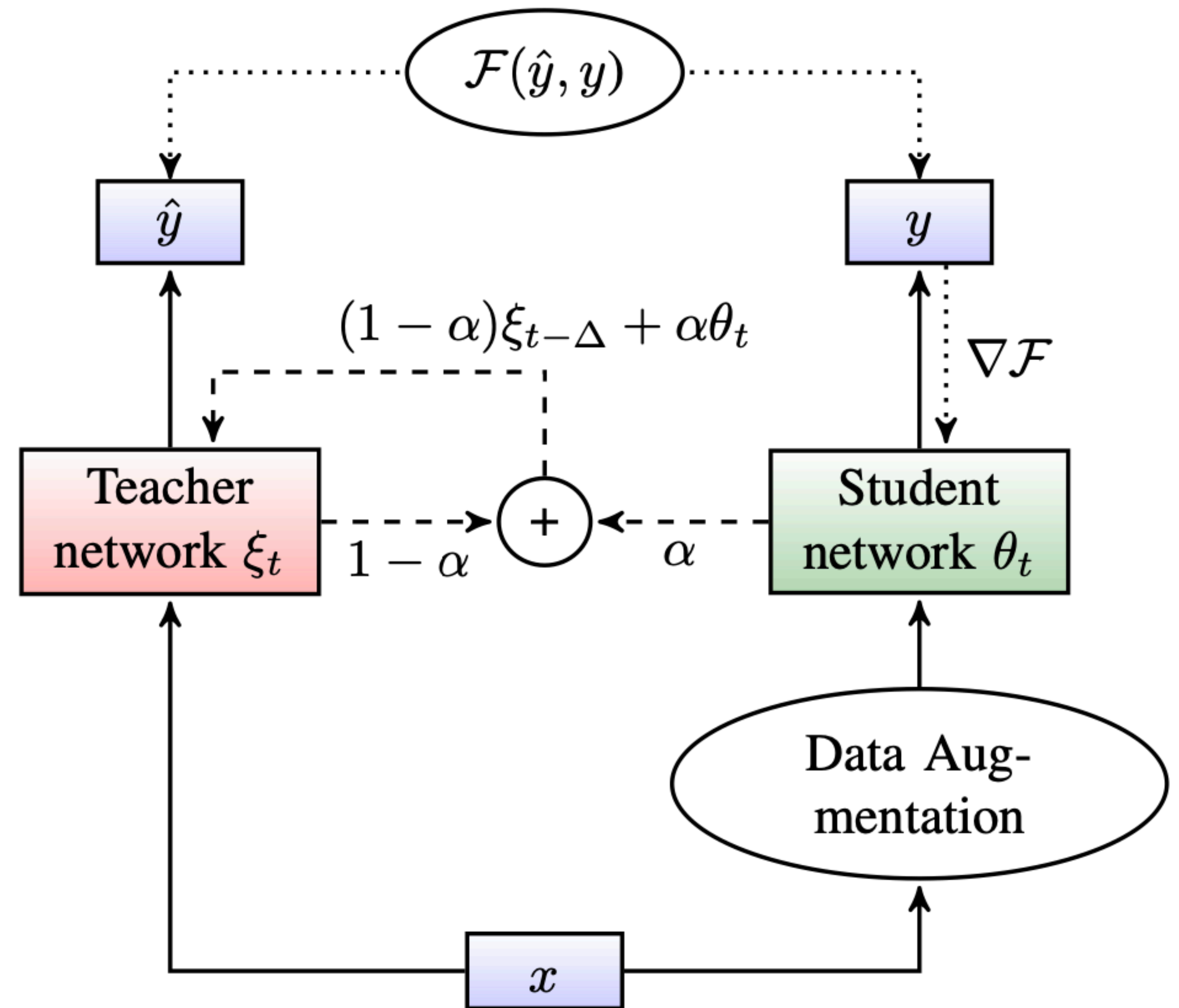
## slimIPL

Table 2: Comparison with other semi- and unsupervised methods: LL-10/LS-960 (top) and LS-100/LS-860 (bottom).

Method	Stride	Tokens	Criterion	LM	Dev WER		Test WER		Compute Resources		
					clean	other	clean	other	Train Time (Days)	# G/TPUs	G/TPU-days
Libri-Light [2]	20 ms	letters	CTC	word 4-gram	30.5	55.8	30.1	57.2	-	-	-
IPL [5]	80ms	5k wp	CTC	- + rescoring	23.8	25.7	24.6	26.5	3	64 GPUs	192
					23.5	25.5	24.4	26.0			
wav2vec 2.0 [28]	20ms	letters	CTC	-	8.1	12.0	8.0	12.1	2.3	128 GPUs	294.4
				word 4-gram	3.4	6.9	3.8	7.3			
				word Transf.	2.9	5.7	3.2	6.1			
slimIPL	30ms	letters	CTC	-	11.4	14	11.4	14.7	4.7	16 GPUs	75.2
				word 4-gram	6.6	9.6	6.8	10.5			
				+ rescoring	5.3	7.9	5.5	9.0			
IPL [5]	80ms	5k wp	CTC	-	5.5	9.3	6.0	10.3	3	64 GPUs	192
				+ rescoring	5.0	8.0	5.6	9.0			
Improved T/S [9]	-	16k wp	S2S	-	4.3	9.7	4.5	9.5	10 × 5	32 TPUs	1600
				LSTM	3.9	8.8	4.2	8.6			
wav2vec 2.0 [28]	20ms	letters	CTC	-	4.6	9.3	4.7	9.0	2.3	128 GPUs	294.4
				word 4-gram	2.3	5.7	2.8	6.0			
				word Transf.	2.1	4.8	2.3	5.0			
slimIPL	30ms	letters	CTC	-	3.7	7.3	3.8	7.5	5.2	16 GPUs	83.2
				word 4-gram	2.8	5.6	3.1	6.1			
				+ rescoring	2.2	4.6	2.7	5.2			

# EMA

- Вместо хранения псевдолейблов от старых версий модели можно усреднять веса с разных версий
- Перебирая параметры alpha и Delta, мы будем больше склоняться к более новым или более старым версиям
- Ищем баланс между расхождением и субоптимальным качеством



**Fig. 1.** Block diagram of the Kaizen framework.

# wav2vec 2.0 + псевдолейблы

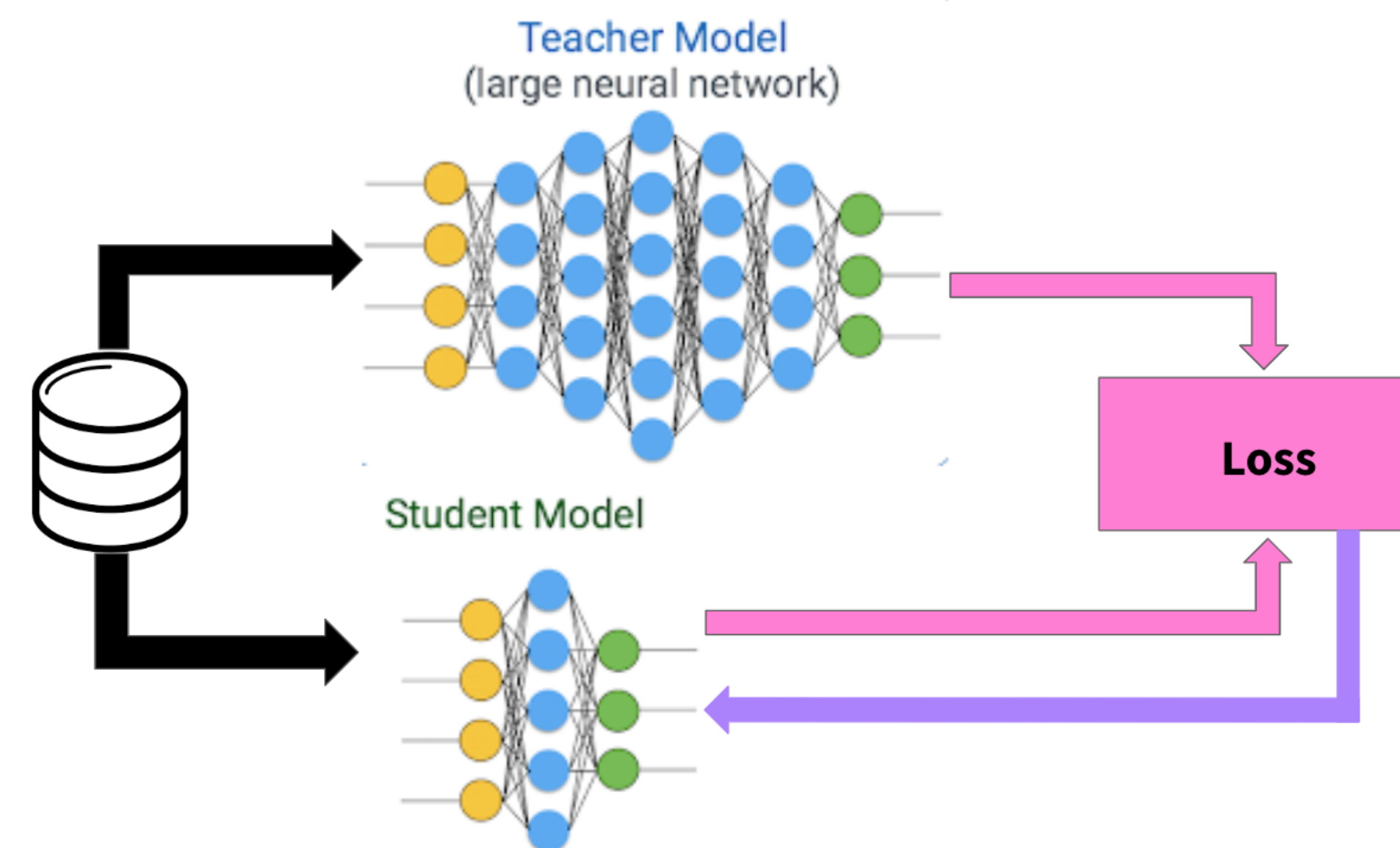
- обучаем wav2vec, фэйн-тюним на размеченном датасете, генерируем псевдолейблы
- особенно полезно, если у нас совсем мало разметки
- fun fact: LAS дополнительно выигрывает от псевдолейблов, т.к. декодер меньше переобучается

Table 3: WER on Librispeech with and without a language model (LM) for 10 min, and 960h of labeled data and LibriVox as unlabeled data.

Model	dev		test	
	clean	other	clean	other
<b>10 min labeled</b>				
wav2vec 2.0 [24]	5.0	8.4	5.2	8.6
- LM	38.3	41.0	40.2	38.7
wav2vec 2.0 + ST (s2s scratch)	2.6	4.7	3.1	5.4
- LM	3.3	5.9	3.7	6.5
wav2vec 2.0 + ST (ctc ft)	2.8	4.6	3.0	5.2
- LM	4.2	6.9	4.3	7.2
<b>960h labeled</b>				
wav2vec 2.0 [24]	1.6	3.0	1.8	3.3
- LM	2.1	4.5	2.2	4.5
wav2vec 2.0 + ST (s2s scratch)	1.1	2.7	1.5	3.1
- LM	1.3	3.1	1.7	3.5
wav2vec 2.0 + ST (ctc ft)	1.6	2.9	1.8	3.3
- LM	1.7	3.6	1.9	3.9

# Soft labels

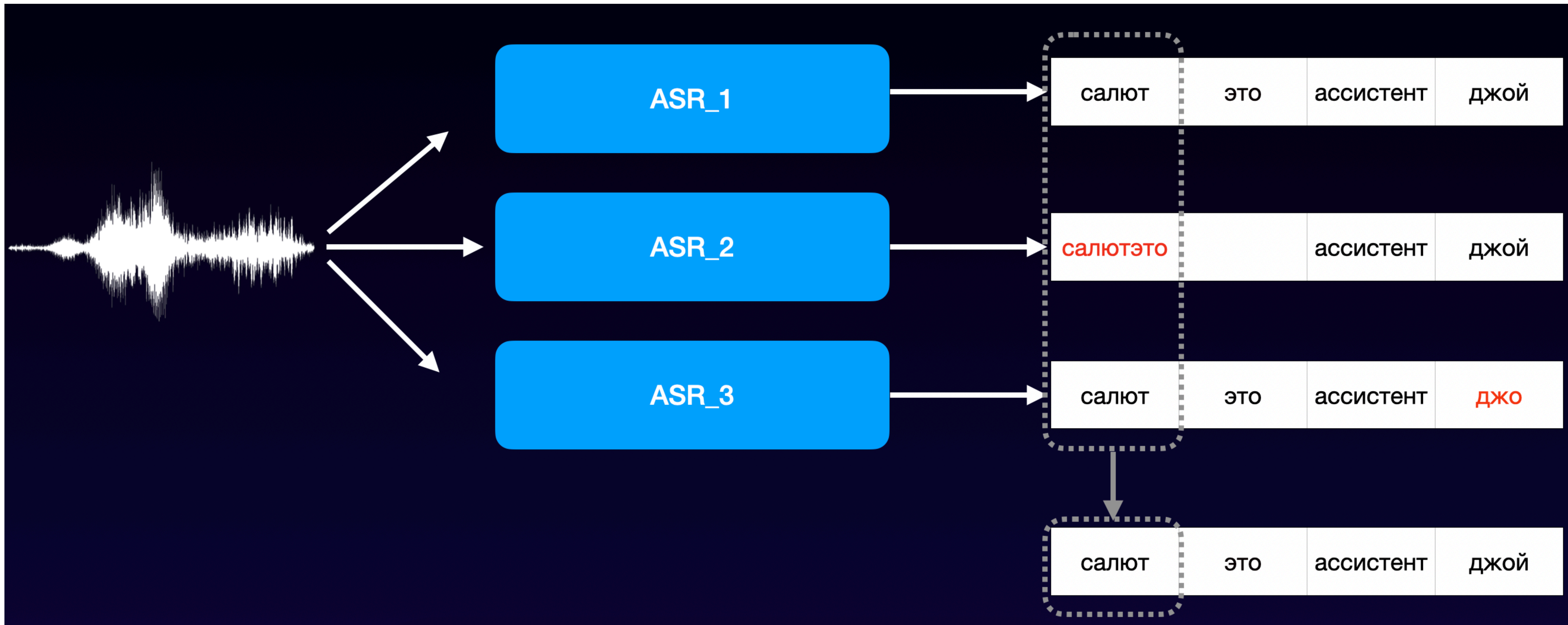
- Мы можем подавать софт-лейблы (иначе говоря логиты) вместо хард-лейблов => получаем по сути классическую дистилляцию
- Соответственно, нужен другой лосс (KL divergence)
- Плохо работает в комбинации с CTC/RNN-T (почему?) — исключение: *self-training* <https://arxiv.org/abs/2210.05793>
- Если стартовать с тех же весов, то лосс может стать равен нулю :) Поэтому аугментации особенно актуальны





# Дистилляция

Используем ансамбль моделей в качестве учителя



# Recap

- Псевдолейблы помогают использовать неразмеченные данные и улучшать качество
- Псевдолейблы дают модели новую информацию за счет:
  - beam search decoding'a
  - языковой модели
  - аугментаций
  - более сильных моделей-учителей (включая ансамбли)
- Делая новые итерации псевдолейблов, мы улучшаем качество еще больше
- Unsupervised pre-training дает дополнительный профит
- Можно использовать софт-лейблы, получая от них дополнительную информацию (плохо работает для CTC)

**Спасибо! Вопросы? :)**